

PASCAL Revision A

- Errata
- Some Helpful Hints...

ERRATA

The following errata identifies page numbers and changes to those pages. Instructions are provided to indicate how the manual pages should be modified. You may choose to make these corrections directly to your manual or simply file this package with your manual.

CHAPTER 3

Page 3-2 First sentence of last paragraph, change to read:

"An expression is composed of operands, operators and function calls, which when evaluated yield a value that can be used for the assignment to an identifier."

CHAPTER 4

Page 4-4 Second to last sentence, change to read:

"It is important to note that an assignment to a value parameter..."

Page 4-6 First sentence of middle paragraph, "...by the called subprogram...", add:

"...so that changes to the parameter actually change the original variable."

Page 4-3

Replace the FORWARD example with the program below:

```

PROGRAM FW (input, output);

VAR
  a, b, c : INTEGER;
  PROCEDURE first(x, y, z:INTEGER);FORWARD;

  PROCEDURE second (i,j,k,l,m:INTEGER);
    BEGIN
      WRITELN ('Entering 2nd with I = ',i);
      IF (i=1) THEN
        first (11, 22, 33)
      ELSE
        first (99, 88, 77);
      END;
    END;

  PROCEDURE first; {(x, y, z:INTEGER);FORWARD;}
    BEGIN
      WRITELN ('Entering 1st with X = ',x);
      IF (x=1) THEN BEGIN
        second(x, y, z, 10, 20)
      END
      ELSE BEGIN
        WRITELN ('Within 1st: X, Y, Z = ',x,y,z);
      END .
    END;

  BEGIN {Main program}
    a := 1; b := 2; c := 3;
    first (a, b, c);
    a := 10; b := 20; c := 30;
    second (a, b, c, 40, 50);
  END

```

You can expect the following output from the above program:

Entering 1st with X =	1		
Entering 2nd with I =	1		
Entering 1st with X =	11		
Within 1st; X, Y, Z =	11	22	33
Entering 2nd with I =	10		
Entering 1st with X =	99		
Within 1st; X, Y, Z =	99	88	77

CHAPTER 5

Page 5-11 Directly below the WITH Statement heading, add:

"The WITH statement can be used to simplify
the process of altering RECORD fields."

CHAPTER 6

Page 6-1 Change the first sentence under the File Type
heading, to read:

"The file type is the only structured
information type that is not fixed by its
definitions and declarations."

APPENDIX D

Page D-1

External:

In version "1.9" of Pascal the external calling sequences do not work. This will be fixed in the next release. If they did work, you would need the information below:

In order to use the "external" option, you have to do a few "extra" things:

- If you plan to use any of the ROM Kernal or AmigaDOS function calls, you need to link with Amiga.lib. Amiga.lib is not currently shipped on the Pascal diskette so you will have to purchase the "C" or Assembler system.
- The Case (upper/lower) of the external calls are very significant to the linker. You must therefore use the CASE switch in the Pascal command line (see example below).
- The external functions use the underscore () character and the EXTERNAL statement. Therefore you need to specify the EXTEND switch in the Pascal command line (see example below).

Example:

```
pascal foo.pas to foo.obj extend case
```

SOME HELPFUL HINTS...

Listed below are some helpful hints for using Pascal. They are provided to supplement the information provided in the Pascal manual.

Using the GOTO Statement

There are restrictions on what statements you can "GOTO". The list below gives you some examples of using the GOTO statement in different ways. It supplements the information given on Page 3-13 and 3-14.

1. The simplest case is that you can go to a statement in the same "statement sequence" that contains the GOTO:

```

PROGRAM SimpleGOTO (input, output);
LABEL 1,2,3,4;
VAR
    x,y,z : CHAR;
    i,j,k : INTEGER;
BEGIN
    i := 3;
    j := 0;
    k := 0;
    1:WRITELN ('Enter A or B');
    READ(x);
    IF (x = 'A') THEN GOTO 2
    ELSE
        IF (x = 'B') THEN GOTO 3
        ELSE
            GOTO 1;

    2:j := i;
      k := -i;
      GOTO 4;

    3:j := -i;
      k := i;

    4:WRITELN ('i,j,k = ',i,j,k);
END.
```

The second valid use of a GOTO statement is where you can "jump out" of a called subroutine (or function) and back into a calling routine:

```

PROGRAM ExitGOTO (input, output);
LABEL 501;
VAR
    x : CHAR;

PROCEDURE called(a: CHAR);
BEGIN
    WRITELN ('The letter you typed is "',a,'"');
    IF (.(a = 'A') or (a = 'B')) THEN
        BEGIN
            WRITE ('You are');
            GOTO 501;
        END
    END;

BEGIN
    WRITELN ('Beginning of "main" program. ');
    WRITELN ('Enter A or B');
    READ(x);
    called (x);
    WRITE ('You are NOT');

    501: WRITELN (' very good at following instructions. ');
END.

```

The third valid use of a GOTO statement is within the statement that contains the GOTO. An example follows:

```
PROGRAM WithinGOTO (input, output);
LABEL 1;
VAR
    x : CHAR;

BEGIN
    WRITELN ('Beginning of "main" program. ');
    WRITELN ('Enter some text and then <Control>-\ (backslash). ');
    WRITELN ('Note this program will not echo text until the ');
    WRITELN ('buffer fills up or end-of-file is found. Also ');
    WRITELN ('<RETURN>s will be turned into <SPACE>s. ');
    1: IF ( NOT EOF ) THEN BEGIN
        READ (x);
        WRITE (x);
        GOTO 1;
    END;
END.
```

Compiling and Executing a Pascal Program

Listed below are a few examples of compiling and executing a Pascal program. They are extensions of the information provided in Appendix F.

1. With the following execute file (S:MAKE) you can type "execute make test1<RETURN>" and have the Pascal program "test1.pas" compile and link.

```
.key F1/a
pascal <F1>.pas to <F1>.obj
alink L:pascal/pstartup.obj+<F1>.obj lib=L:pascal/paslib.obj
to <F1>
```

Note that by not using .WITH files for the alink command you gain flexibility IF the alink commands for all programs look the same.

2. If most programs you'll be compiling have different alink command lines the following would be a better choice:

```
.key FILE/a
pascal <FILE>.pas to <FILE>.obj
alink with <FILE>.with
```

And the WITH file might be:

```
FROM L:pascal/startup.obj*
    myprogram.obj
LIBRARY L:pascal/paslib.obj
TO myprogram
```

Note that each WITH program will have different file names in it and that you cannot use the <FILE> style execute substitutions.

Linking External Routines

LIST Option

The information below is provided to supplement the LIST Option information found in Appendix F on Page F-1

1. This version is shipped on a V1.0 "CLI" (system) diskette. As such it includes printer drivers and other libraries. When you get version 1.1 of Workbench or CLI you should update the system files, creating a new Pascal diskette.
2. Because there is a bug in the V1.0 software, the PRT: device cannot be loaded at the same time as PAR: or SER:. What this means is that you must output to PAR: or SER: (depending on how your preferences are setup) before you output to prt:. This will be fixed in V1.1.
3. Given #2 above, you can use one of the three methods below of initializing and listing to the printer. The first two initialize the parallel port, then print to the printer. The third outputs to a disk file first, then initializes the serial port and outputs to the printer. SER: and PAR: may be exchanged freely in the following examples, depending on your Preferences setting.

```
date > par:
pascal foo.pas to foo.obj list PRT:
```

or

```
copy * par:
<RETURN>
<Control>- \
pascal foo.pas to foo.obj list PRT:
```

or

```
pascal foo.pas to foo.obj list print_file
date > ser:
copy print_file prt:
```

WS Option

The information below is provided to supplement the WS Option information found in Appendix F on Page F-1

1. In order to use the work space option, type (for example):

```
pascal foo.pas to foo.obj ws 50
```

The 50 in this example represents 50K (51,200) bytes.
The default is 25K (not 30K as documented).



**This was brought to you
from the archives of**

<http://retro-commodore.eu>